

A Rule-Based Co-Adaptive Regulator for Keeping Novice Programmers in An Optimal Difficulty Corridor: Design of The Algoadapt System and A Simulation Study

 Nazokat R. Kayumova

Jizzakh Branch of the National University of Uzbekistan named after Mirzo Ulugbek, Jizzakh, Uzbekistan

Received: 12 March 2026; **Accepted:** 08 April 2026; **Published:** 31 May 2026

Abstract: Background. Introductory programming courses enrol students whose prior experience differs widely, yet most practice sequences present the same tasks to everyone. Adaptive systems can individualise difficulty, but many rely on opaque models that teachers neither understand nor control. **Purpose.** This study designs and verifies an explainable adaptation mechanism for developing first-year students' algorithmic competence. **Methods.** Following a design-and-development research approach, we built AlgoAdapt, a web-based system with a two-loop (co-adaptive) architecture: an inner loop in which a five-rule regulator (Q1–Q5) sets task difficulty and the amount of help after every attempt, and an outer loop in which teachers act on a process-based learner model of four competence components (cognitive, practical, metacognitive, reflective). The regulator was verified in a Monte Carlo simulation with 2,000 synthetic learners per condition and compared with an ablated version without hysteresis, a 1-up/1-down staircase and a fixed linear sequence. **Results.** After a short convergence period (median five tasks), the regulator kept 92.2% of tasks within the target difficulty corridor, compared with 68.2%, 61.6% and 19.3% for the three comparison policies, and produced on average 0.62 direction reversals per 60 tasks versus 34.67 for the staircase. The advantage held when ability grew during practice and across three values of the response-model slope. **Conclusions.** A small set of transparent, hysteresis-protected rules that first increases help and only then lowers difficulty can track learners stably. Classroom evaluation of learning outcomes is the necessary next step.

Keywords: Adaptive learning; algorithmic competence; introductory programming; explainable adaptation; zone of proximal development; scaffolding; learning analytics; design-based research.

1. Introduction: Learning to construct algorithms and express them in a programming language is a known bottleneck of computing curricula. Multi-institutional studies have repeatedly shown that, at the end of their first course, many students cannot design, trace or explain even simple programs (Lister et al., 2004; McCracken et al., 2001), and reviews of introductory programming attribute part of the difficulty to the simultaneous load of syntax, semantics, problem decomposition and debugging strategies (Luxton-Reilly et al., 2018; Robins et al., 2003). Current curriculum frameworks therefore describe the target not as

knowledge of a language but as a competence that integrates knowledge, skills and dispositions (ACM & IEEE Computer Society, 2020).

A second, structural difficulty is the heterogeneity of first-year cohorts. In Uzbek universities, as elsewhere, graduates of specialised lyceums who have already programmed sit in the same group as students who meet loops and arrays for the first time. A uniform sequence of exercises is then too easy for the first group and too hard for the second, so that both spend much of their practice time outside what Vygotsky (1978) called the zone of proximal development: the

range of tasks a learner cannot yet perform alone but can perform with support. Cognitive load theory adds that the optimal level of guidance itself depends on expertise; support that helps a novice can hinder an advanced learner (Kalyuga, 2007; Sweller et al., 2011).

Adaptive and intelligent tutoring systems address this problem by individualising task selection and feedback, and meta-analyses report substantial benefits compared with conventional instruction (Kulik & Fletcher, 2016; VanLehn, 2011). Automated assessment of programming assignments is mature (Ihantola et al., 2010), and automated feedback has been reviewed extensively (Keuning et al., 2018). However, two gaps remain relevant to everyday university teaching. First, much adaptive logic is implemented as statistical models such as knowledge tracing (Corbett & Anderson, 1994) or Elo-type ratings (Klinkenberg et al., 2011; Pelánek, 2016), whose decisions are difficult for teachers to inspect. Recent analyses of AI in education argue that the field is moving towards co-adaptive, human-centred designs in which teachers remain in control and understand why a system acts (Feng et al., 2025; Hooshyar et al., 2025). Second, adaptation is usually framed as choosing which item comes next; much less attention is paid to how much help accompanies it, although scaffolding research treats contingent and fading help as a central mechanism of learning (Wood et al., 1976).

Treating adaptation as a feedback-control problem offers a useful lens on both gaps. A controller measures the state of a process, compares it with a set point and applies a corrective action; good controllers are stable and do not oscillate (Åström & Murray, 2008). In instruction, the “process” is the learner’s interaction with tasks, the set point is a difficulty level within the zone of proximal development, and the corrective actions are changes in difficulty and in support. Work on optimal training difficulty suggests that learning is fastest when success rates are moderately high rather than maximal (Wilson et al., 2019), which again implies a target band rather than a single difficulty value.

This article reports the design of AlgoAdapt, an adaptive system for first-year programming courses that implements this control view with a small set of explicit rules, and a simulation study that verifies whether the rules behave as intended before they are evaluated with learning outcomes in classrooms. The

study addresses three research questions:

RQ1. How can an adaptive system for developing algorithmic competence be structured so that automatic adaptation remains transparent to, and controllable by, the teacher?

RQ2. Does a rule-based regulator that adjusts both task difficulty and the amount of help keep learners with different prior ability within a target difficulty corridor, and how quickly does it converge?

RQ3. How do the hysteresis condition and the “help first, difficulty second” order contribute to the stability of the regulator compared with simpler policies?

2. METHODS

2.1. Research approach

The study follows design-and-development research, in which an instructional product is designed on theoretical grounds, implemented and evaluated in iterative cycles (McKenney & Reeves, 2019; Richey & Klein, 2007; Wang & Hannafin, 2005). This article covers the design and implementation phase and an internal verification of the adaptation mechanism. Verification by simulation was chosen as a first step for two reasons: it isolates the behaviour of the rules from other classroom factors, and it allows comparison with alternative policies under controlled conditions that would be impractical or unethical to impose on students. Simulation does not provide evidence of learning gains; such evidence requires a classroom study, which is discussed in Section 4.

2.2. Design requirements

Six requirements were derived from the literature reviewed above and from an analysis of teaching practice in first-year programming courses: (a) adaptation of task difficulty to each learner’s current state; (b) assessment based on process evidence, not only on final answers (Pellegrino et al., 2001); (c) feedback at the task, process and self-regulation levels, avoiding person-level comments (Hattie & Timperley, 2007; Shute, 2008); (d) graduated, fading scaffolding (Wood et al., 1976); (e) explainability of every automatic decision; and (f) a teacher role that is strengthened rather than replaced (Feng et al., 2025; Hooshyar et al., 2025). Algorithmic competence was operationalised as four components: cognitive (knowledge of algorithmic concepts and structures),

practical (designing, coding and testing algorithms), metacognitive (planning, monitoring and choosing strategies) and reflective (analysing errors and improving one's own solution), drawing on Zimmerman's (2002) cycle of self-regulated learning.

2.3. The AlgoAdapt system

AlgoAdapt 2.0 is a web application (Python, FastAPI, SQLite) registered with the Ministry of Justice of the Republic of Uzbekistan as computer software (certificate No. DGU 64245, 14 May 2026). Content is organised into seven modules that follow the first-year sequence "Programming 1–2": linear algorithms, branching, loops, arrays and strings, functions and modular programming, recursion, and sorting and searching. Each module has ten difficulty sub-levels grouped into three bands (basic 1–3, intermediate 4–7, advanced 8–10) and 15 parametric variants per sub-level, giving a bank of 1,050 tasks. Code tasks are checked against test cases by a local judge for Python, C++, C# and Java; the remaining tasks are short-answer items. Each task carries three graded hints (a guiding question, a plan or pseudocode fragment, and a key part of the solution) and a reflection prompt (e.g.,

"Which strategy did you choose, and how did you find the cause of the error?").

The system is built around two loops (Figure 1), following the co-adaptive paradigm described by Feng et al. (2025). In the **inner loop**, which runs after every submission, a process sensor records the result, the attempt number, the ratio of solution time to the task's normative time, the error type (compilation, runtime, wrong answer, time-out), the hint level used and the text of the reflection. The learner model is updated and the regulator chooses the difficulty and the amount of help for the next task; the rule that fired is stored with the attempt so that every decision can be traced. In the **outer loop**, which runs at the scale of lessons and weeks, the teacher dashboard shows component profiles, the most frequent error types and students at risk. Teachers plan group analysis of typical errors, work individually with struggling students and may override a student's level; overrides are logged with a note. Teachers can also edit tasks, hints and feedback messages, which changes the behaviour of the inner loop.

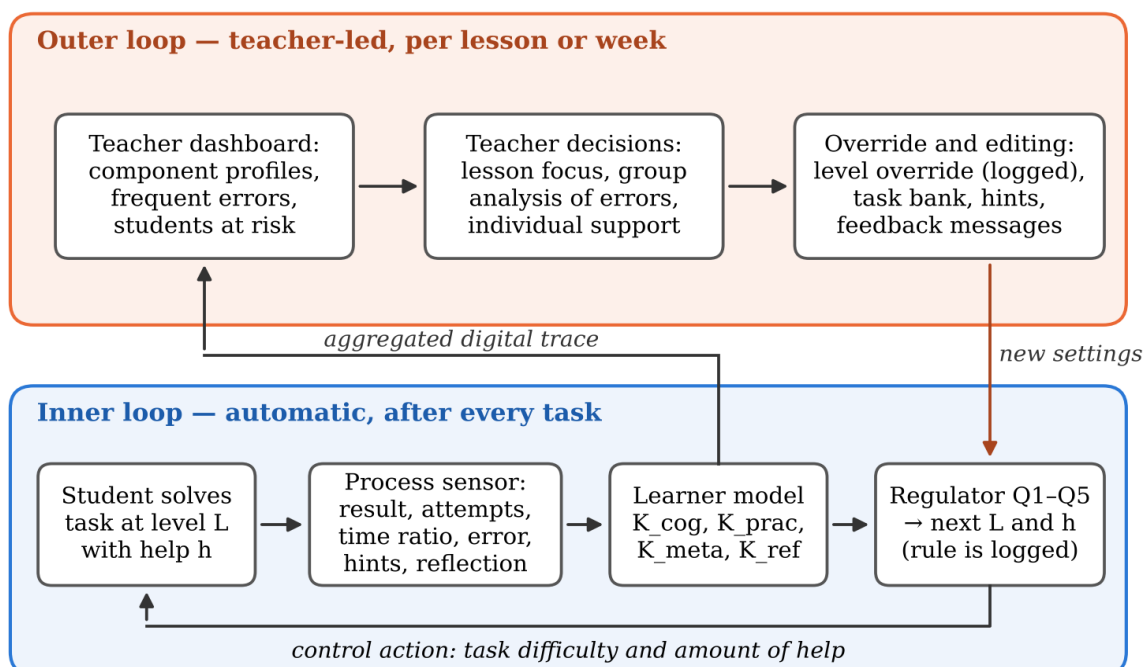


Figure 1. Two-loop (co-adaptive) architecture of AlgoAdapt. The inner loop adapts difficulty and help after every task; the outer loop lets the teacher act on aggregated process data and adjust the system.

2.4. Process-based learner model

For each student and module, four component indicators are computed from the last 40 attempts.

Each indicator is a weighted sum of normalised process measures:

$$K_j = \sum w_i \cdot x_i, \quad \sum w_i = 1, \quad x_i \in [0, 1] \quad (1)$$

where x_i are the measures listed in Table 1 and w_i their weights. The integral indicator of algorithmic competence is the mean of the four components:

$$I = (K_{\text{cog}} + K_{\text{prac}} + K_{\text{meta}} + K_{\text{ref}}) / 4 \quad (2)$$

Values of $K \geq 0.75$ are interpreted as a high level, $0.50 \leq K < 0.75$ as intermediate and $K < 0.50$ as low.

Reflection quality is estimated by a transparent heuristic (length of the answer and presence of strategy-related terms such as “error”, “cause”, “test”), which teachers can review in the dashboard. The weights are design choices informed by the requirements above and are intended to be calibrated against expert ratings in later cycles.

Table 1. Process measures and weights used to compute the competence components in AlgoAdapt 2.0

Component	Process measures (x_i) and weights (w_i)
Cognitive, K_{cog}	share of correct answers (0.65); mean score (0.35)
Practical, K_{prac}	mean score (0.55); share of correct answers (0.25); 1 – normalised number of repeated attempts (0.20)
Metacognitive, K_{meta}	share of attempts solved without hints (0.55); 1 – normalised overrun of normative time (0.45)
Reflective, K_{ref}	mean reflection quality (0.60); share of tasks in which an initial error was corrected by the student (0.40)

Note. Measures are computed over the last 40 attempts of the student in the module; all values are bounded to $[0, 1]$.

2.5. The difficulty regulator

The regulator controls two variables: the difficulty sub-level L (1–10) and the help level h (0–3). Its goal is to keep the learner in what we call the **optimal difficulty corridor**: tasks that are neither solved instantly without effort nor failed repeatedly even with help. The lower

edge of the corridor is signalled by a fast, first-attempt, unaided success; the upper edge by repeated failure despite maximal help. Inside the corridor, success with some effort or some help leads to consolidation at the same level. Table 2 lists the five rules exactly as implemented in AlgoAdapt 2.0.

Table 2. Adaptation rules of the AlgoAdapt regulator

Rule	Condition (from the attempt log)	Action
Q1	Correct on the first attempt, time ratio < 0.85 , no hints, and this is the second such success in a row	Raise difficulty by one sub-level; reset help
Q2	Correct, but with hints, more than one attempt, slower than 0.85 of the normative time, or first fast success only	Keep the level (consolidation); reduce help by one step
Q3	Incorrect and help level < 3	Keep the level; increase help by one step
Q4	Incorrect with maximal help, second consecutive failure, level > 1	Lower difficulty by one sub-level; help level set to 2
Q5	Incorrect with maximal help, otherwise (first failure at maximal help, or level 1)	Keep level and maximal help; recommend review and reflection

Note. Time ratio = solution time / normative time of the task ($90 + 20 \cdot L$ seconds). Moving to the next module is initiated by the student or the teacher.

Two design decisions distinguish the regulator from a simple up-down rule. First, **hysteresis**: difficulty rises only after two consecutive fast, unaided successes (Q1) and falls only after two consecutive failures at maximal help (Q4), so that a single lucky or unlucky attempt does not change the level. Second, **help first, difficulty second**: when a learner fails, the system increases support step by step (Q3) and lowers difficulty only when support has been exhausted (Q4). This order follows the contingent-scaffolding principle (Wood et al., 1976) and encourages students to persist with tasks at their level. When help is used and the task is solved, help is withdrawn gradually (Q2), implementing fading.

2.6. Simulation study

Learner model. Each synthetic learner has a latent ability θ expressed on the same 1–10 scale as the sub-levels. The probability of a correct answer on a task of level L with help level h follows a logistic response function:

$$P(\text{correct}) = 1 / (1 + \exp[-(a(\theta - L + b) + c \cdot h)]) \quad (3)$$

with slope $a = 1.2$, offset $b = 0.5$ (a learner succeeds unaided with probability 0.65 when $L = \theta$) and help effect $c = 0.5$ per hint level. The time ratio was drawn as $\exp[0.3(L - \theta) + \epsilon]$, $\epsilon \sim N(0, 0.3^2)$, so that tasks below ability are solved faster than the norm. Initial ability was drawn uniformly from $[1, 9]$; every learner started at level 1 with no help, as in the deployed system.

Policies. Four policies were compared on identical learners and random streams: (1) the AlgoAdapt regulator (Table 2), with the function copied from the production code; (2) an ablation without hysteresis, in which a single fast unaided success triggers Q1; (3) a 1-up/1-down staircase without help, a classic adaptive procedure (Levitt, 1971); and (4) a fixed linear sequence that advances one sub-level every six tasks, representing a common non-adaptive syllabus.

Scenarios and measures. Each policy was run for 60 tasks with 2,000 learners in two scenarios: stable ability, and ability growing by 0.05 sub-levels per task (three sub-levels over the run), which tests whether the regulator can track a moving target. A task was counted as in the corridor when the learner's unaided success probability lay between 0.35 and 0.85, i.e., when the task was neither trivial nor out of reach. We report the

share of tasks in the corridor (over all 60 tasks and over tasks 21–60, after convergence), accuracy, the number of direction reversals of the level (a measure of oscillation), the convergence time (first task after which the level stays within ± 1 of ability in at least five of six tasks) and the frequency of each rule. Sensitivity to the response model was examined with $a \in \{0.8, 1.2, 1.6\}$ (1,000 learners each). The simulation code is provided as supplementary material, and all random seeds are fixed so that the results can be reproduced exactly.

3. RESULTS

3.1. Architecture and transparency (RQ1)

The implemented system realises the six requirements in the following way. Adaptation and process-based assessment are carried by the sensor, learner model and regulator of the inner loop; feedback is given at the task level (which tests failed), the process level (error type and its likely cause) and the self-regulation level (a question prompting the student to revise the strategy), with three graded hints. Explainability is achieved because every adaptive decision is produced by one of five readable rules and stored with the attempt (the attempt log contains the rule, the level before and after, the help level, the time and the error type). Teacher control is ensured by the outer loop: the dashboard, logged overrides, editable task bank and hints, and CSV export of all attempts for further analysis. In this design the system does not replace pedagogical judgement; it supplies the teacher with structured evidence and executes routine micro-decisions that a teacher could not make for every student after every task.

3.2. Behaviour of the regulator (RQ2)

Table 3 summarises the simulation. With stable ability, the AlgoAdapt regulator converged after a median of five tasks and then kept 92.2% of tasks within the corridor; over the whole run, including the initial climb from level 1, the share was 84.3%. Accuracy was 66.9%, which corresponds to moderately challenging practice. The level changed direction on average 0.62 times per 60 tasks, i.e., most learners moved monotonically towards their level and then stayed there. Only 0.5% of learners had not converged by the end of the run.

Table 3. Simulation results for four adaptation policies (2,000 synthetic learners per policy, 60 tasks each)

Policy	In corridor, all tasks, % (SD)	In corridor, tasks 21– 60, %	Accuracy, %	Reversals, M (SD)	Convergence, median tasks
Stable ability					
AlgoAdapt (Q1–Q5)	84.3 (16.3)	92.2	66.9	0.62 (0.86)	5
Without hysteresis	67.4 (24.7)	68.2	59.2	1.82 (1.36)	4
1-up/1-down staircase	58.8 (10.4)	61.6	53.8	34.67 (3.65)	7
Fixed linear sequence	19.6 (1.9)	19.3	50.5	0.00 (0.00)	23
Growing ability (+0.05 per task)					
AlgoAdapt (Q1–Q5)	81.5 (11.9)	89.9	74.5	0.27 (0.71)	5
Without hysteresis	81.1 (11.6)	86.1	66.3	0.85 (1.18)	4
1-up/1-down staircase	61.0 (7.1)	64.6	60.0	32.75 (5.65)	7
Fixed linear sequence	22.5 (9.8)	24.7	66.6	0.00 (0.00)	29

Note. Corridor = unaided success probability between 0.35 and 0.85. Reversals = number of changes in the direction of level movement. In the growing-ability scenario, 11.7% of learners never converged under the fixed linear sequence; the corresponding share was at most 1.4% for all other policies in both scenarios.

Figure 2 shows typical trajectories. A learner with low prior ability ($\theta = 2.5$) reached level 2 after four tasks and level 3 later in the run; a learner of medium ability ($\theta = 5.0$) settled at level 5 after seven tasks and moved to level 6 only near the end of the run; a high-ability learner ($\theta = 8.0$) climbed to level 9 within 14 tasks. Because Q1 requires two consecutive fast successes, advanced students need about two tasks per sub-level

to climb, which explains the longer convergence for the high-ability third of the sample (median seven tasks, compared with one and four tasks for the low- and medium-ability thirds). After convergence, the share of tasks in the corridor was almost identical across the three ability groups (93.9%, 91.4% and 91.5%), indicating that the regulator serves weak and strong students equally well.

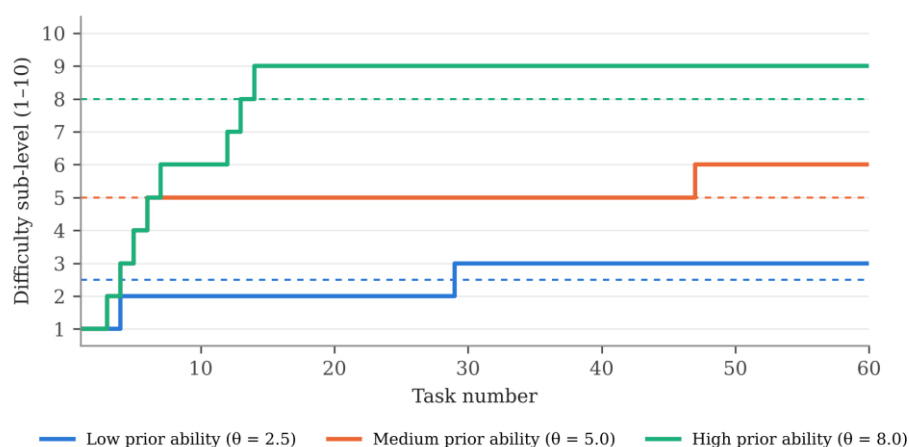


Figure 2. Difficulty trajectories produced by the AlgoAdapt regulator for three synthetic learners with low, medium and high prior ability (stable-ability scenario). Dashed lines show the learners' latent ability.

When ability grew during practice, the regulator still kept 89.9% of later tasks in the corridor, and accuracy rose to 74.5% because learners were often slightly ahead of the current level before two qualifying successes moved them up. Rule frequencies show how the corridor was maintained: in the stable scenario, consolidation (Q2) accounted for 58.7% of decisions, increases of help (Q3) for 32.3%, increases of difficulty (Q1) for 8.2% and decreases of difficulty (Q4) for only 0.8%. In the growing-ability scenario the corresponding values were 63.1%, 25.2%, 11.4% and 0.2%. Thus, difficulty was lowered very rarely: most difficulties were resolved by help, which was then withdrawn.

3.3. Contribution of hysteresis and of the help-first order (RQ3)

Removing hysteresis made the regulator react faster (median convergence four tasks instead of five) but less stable: reversals tripled (1.82 vs. 0.62) and the share of later tasks in the corridor fell from 92.2% to 68.2% with

stable ability (Table 3, Figure 3). Without hysteresis, a single fast success at a level slightly below ability was enough to move the learner up, after which failures pushed help to its maximum; learners therefore spent more time at the edge of their ability. The difference was smaller when ability was growing (89.9% vs. 86.1%), because then rapid upward movement was partly justified.

The staircase, which changes difficulty after every response and offers no help, oscillated continuously (about 33–35 reversals per 60 tasks) and kept only 61.6–64.6% of later tasks in the corridor. The fixed linear sequence did not oscillate but was mismatched to most learners: low-ability learners soon faced tasks far above their level (accuracy 24.2%), and high-ability learners spent most of the run on tasks far below it (accuracy 75.7% with stable ability, 94.2% with growing ability). Its share of tasks in the corridor did not exceed 25%.

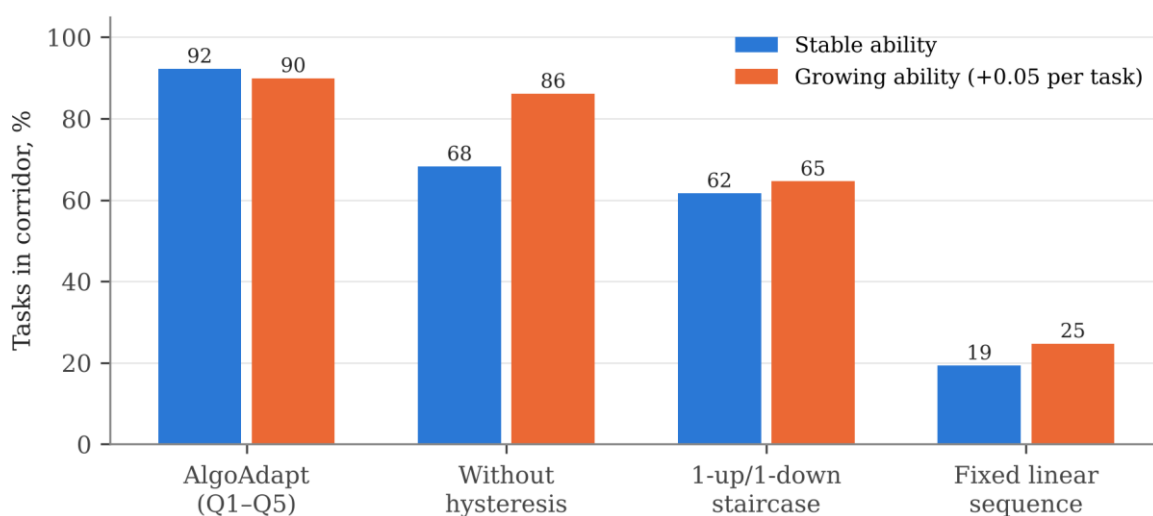


Figure 3. Share of tasks 21–60 that fell within the optimal difficulty corridor under four adaptation policies, for stable and growing learner ability (2,000 synthetic learners per bar).

The ranking of policies was the same for all three values of the response-model slope (Table 4). The regulator's share of later tasks in the corridor ranged from 87.9% to 97.2%, while the best comparison policy reached

64.6% to 79.8%. The steeper the slope (i.e., the sharper the boundary between tasks a learner can and cannot do), the harder the tracking problem becomes for all policies, but the regulator retained the largest margin.

Table 4. Sensitivity of the share of tasks 21–60 in the corridor (%) to the slope a of the response model (stable ability, 1,000 learners per cell)

Policy	$a = 0.8$	$a = 1.2$	$a = 1.6$
AlgoAdapt (Q1–Q5)	97.2	92.5	87.9
Without hysteresis	79.8	68.6	64.6

Policy	a = 0.8	a = 1.2	a = 1.6
1-up/1-down staircase	71.6	61.3	53.4
Fixed linear sequence	29.0	20.0	15.4

4. DISCUSSION

The study set out to design an adaptive system for developing algorithmic competence whose adaptation is transparent to teachers, and to verify the behaviour of its adaptation mechanism before classroom evaluation. Regarding RQ1, the two-loop architecture shows how the co-adaptive, human-centred paradigm described by Feng et al. (2025) and Hooshyar et al. (2025) can be made concrete in an ordinary university course: the automatic loop handles frequent micro-decisions, while the teacher loop handles interpretation, motivation and the improvement of content. Because the regulator consists of five readable rules and every decision is logged with the rule that produced it, a teacher can reconstruct why a student received a given task, a property that statistical learner models usually lack (Pelánek, 2016).

Regarding RQ2, the simulation shows that simple rules are sufficient to keep learners of very different ability within a target band of difficulty after a short climb. The resulting accuracy of about 67–75% lies in the range that research on optimal training difficulty associates with efficient learning (Wilson et al., 2019), and it is consistent with the idea of working within, rather than below or above, the zone of proximal development (Vygotsky, 1978). It is worth emphasising that this result concerns the placement of tasks, not learning itself; whether students who spend more time in the corridor learn more must be tested empirically.

Regarding RQ3, the comparison clarifies why the two distinctive design decisions matter. Hysteresis trades a slightly slower start for much greater stability, a classic trade-off in feedback control (Åström & Murray, 2008). The help-first order meant that difficulty was lowered in fewer than 1% of decisions; difficulties were usually resolved by graduated hints that were later withdrawn. Pedagogically, this corresponds to contingent scaffolding with fading (Wood et al., 1976) and avoids a pattern in which learners are moved down at the first sign of difficulty and lose the opportunity to struggle

productively. The staircase result illustrates the opposite risk: a policy that reacts to every response produces constant oscillation, which in a classroom would be experienced as arbitrary changes of task difficulty.

The design also has implications for assessment. Because the inner loop records attempts, time, errors, hints and reflections, the learner model can estimate metacognitive and reflective components that final scores do not capture, in line with an evidence-centred view of assessment (Pellegrino et al., 2001). At the same time, these indicators are only as valid as their operationalisation; the reflection-quality heuristic in particular is a coarse proxy and should be validated against teacher ratings.

4.1. Limitations

The main limitation is that the results come from a simulation, not from students. The learner model assumes a single latent ability per module, a logistic response function and a fixed effect of hints; real students also differ in motivation, guess, copy solutions or use generative AI tools, and their ability changes in ways that depend on instruction. The corridor boundaries (0.35–0.85) and the hint effect are modelling choices, although the sensitivity analysis showed that the ranking of policies did not depend on the response slope. The fixed linear baseline is a simplified representation of a syllabus. Finally, the weights of the learner model have not yet been calibrated against independent measures of competence.

4.2. Implications and future work

For designers of adaptive systems in programming education, the findings suggest three practical principles: adapt both difficulty and support, increase support before lowering difficulty, and require consistent evidence (hysteresis) before changing the level. For teachers, an explainable rule set and a logged override give a basis for trusting and correcting automatic decisions. The next cycle of the design-based

research will evaluate AlgoAdapt in first-year programming courses with a quasi-experimental design, measuring the four competence components with independent instruments, comparing the observed share of tasks in the corridor with the simulated values and calibrating the learner-model weights against expert ratings.

5. CONCLUSION

This article presented AlgoAdapt, an adaptive system for developing first-year students' algorithmic competence built on a two-loop, co-adaptive architecture, a process-based learner model with cognitive, practical, metacognitive and reflective components, and an explainable five-rule regulator of task difficulty and help. A simulation with synthetic learners showed that the regulator converges quickly, keeps more than 90% of tasks within an optimal difficulty corridor for learners of low, medium and high prior ability, and is far more stable than an up-down staircase, with hysteresis and the help-first order both contributing to this stability. These results establish that the adaptation mechanism behaves as designed; its effect on students' learning remains to be demonstrated in classroom studies.

Declarations

Funding. This research received no specific grant from any funding agency.

Conflict of interest. The author is the developer and copyright holder of the AlgoAdapt software (certificate No. DGU 64245).

Ethics. The simulation study did not involve human participants or personal data.

Data and code availability. The simulation code (Python) and the resulting data are available as supplementary material and from the author on request.

Use of generative AI. [The author should describe here, according to the target journal's policy, whether and how AI tools were used in preparing the manuscript.]

REFERENCES

1. ACM & IEEE Computer Society. (2020). Computing Curricula 2020: Paradigms for global computing education. ACM. <https://doi.org/10.1145/3467967>
2. Åström, K. J., & Murray, R. M. (2008). Feedback systems: An introduction for scientists and engineers. Princeton University Press.
3. Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253–278. <https://doi.org/10.1007/BF01099821>
4. Feng, S., Zhang, H., & Gašević, D. (2025). Mapping the evolution of AI in education: Toward a co-adaptive and human-centered paradigm. *Computers and Education: Artificial Intelligence*, 9, 100513. <https://doi.org/10.1016/j.caeai.2025.100513>
5. Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research*, 77(1), 81–112. <https://doi.org/10.3102/003465430298487>
6. Hooshyar, D., Šir, G., Yang, Y., Kikas, E., Hämäläinen, R., Kärkkäinen, T., Gašević, D., & Azevedo, R. (2025). Towards responsible AI for education: Hybrid human-AI to confront the elephant in the room. *Computers and Education: Artificial Intelligence*, 9, 100524. <https://doi.org/10.1016/j.caeai.2025.100524>
7. Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (pp. 86–93). ACM. <https://doi.org/10.1145/1930464.1930480>
8. Kalyuga, S. (2007). Expertise reversal effect and its implications for learner-tailored instruction. *Educational Psychology Review*, 19(4), 509–539. <https://doi.org/10.1007/s10648-007-9054-3>
9. Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education*, 19(1), Article 3. <https://doi.org/10.1145/3231711>
10. Klinkenberg, S., Straatemeier, M., & van der Maas, H. L. J. (2011). Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, 57(2), 1813–1824. <https://doi.org/10.1016/j.compedu.2011.02.003>

11. Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: A meta-analytic review. *Review of Educational Research*, 86(1), 42–78. <https://doi.org/10.3102/0034654315581420>
12. Levitt, H. (1971). Transformed up-down methods in psychoacoustics. *The Journal of the Acoustical Society of America*, 49(2B), 467–477. <https://doi.org/10.1121/1.1912375>
13. Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), 119–150. <https://doi.org/10.1145/1041624.1041673>
14. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55–106). ACM. <https://doi.org/10.1145/3293881.3295779>
15. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4), 125–180. <https://doi.org/10.1145/572139.572181>
16. McKenney, S., & Reeves, T. C. (2019). *Conducting educational design research* (2nd ed.). Routledge.
17. Pelánek, R. (2016). Applications of the Elo rating system in adaptive educational systems. *Computers & Education*, 98, 169–179. <https://doi.org/10.1016/j.compedu.2016.03.017>
18. Pellegrino, J. W., Chudowsky, N., & Glaser, R. (Eds.). (2001). *Knowing what students know: The science and design of educational assessment*. National Academy Press.
19. Richey, R. C., & Klein, J. D. (2007). *Design and development research: Methods, strategies, and issues*. Lawrence Erlbaum Associates.
20. Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and teaching programming: A review and discussion*. *Computer Science Education*, 13(2), 137–172. <https://doi.org/10.1076/csed.13.2.137.14200>
21. Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, 78(1), 153–189. <https://doi.org/10.3102/0034654307313795>
22. Sweller, J., Ayres, P., & Kalyuga, S. (2011). *Cognitive load theory*. Springer. <https://doi.org/10.1007/978-1-4419-8126-4>
23. VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4), 197–221. <https://doi.org/10.1080/00461520.2011.611369>
24. Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.
25. Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research and Development*, 53(4), 5–23. <https://doi.org/10.1007/BF02504682>
26. Wilson, R. C., Shenhav, A., Straccia, M., & Cohen, J. D. (2019). The Eighty Five Percent Rule for optimal learning. *Nature Communications*, 10, 4646. <https://doi.org/10.1038/s41467-019-12552-4>
27. Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2), 89–100. <https://doi.org/10.1111/j.1469-7610.1976.tb00381.x>
28. Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory Into Practice*, 41(2), 64–70. https://doi.org/10.1207/s15430421tip4102_2